**Addressing modes:**

* The 8051 microprocessor supports various addressing modes that enable efficient data access and manipulation.
* These modes define how memory or registers are referenced in instructions, optimizing performance and simplifying programming.

**What is Addressing Mode?**

* An addressing mode in a microprocessor defines the way in which the operand (data) is specified for an instruction.
* It determines how the data is accessed from memory or registers and how it is operated upon.
* Different addressing modes provide flexibility, helping to optimize performance, save memory, and simplify code.

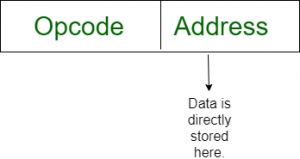
**Types of Addressing Modes:**

**1. Immediate Addressing Mode**

**Definition:**

* In this mode, the operand (data) is directly specified in the instruction using the # symbol.
* The data is stored in the instruction itself.
* If the data is 8-bit, the instruction takes **2 bytes**.
* If the data is 16-bit, the instruction takes **3 bytes**.

**Syntax:**  
MOV A, #data



**Examples:** MOV A, #25H → Load value 25H directly into accumulator A

**Use Case:** Used when a fixed value needs to be assigned to a register or memory location.

**2. Register Addressing Mode**

**Definition:**

* The data is stored in a register, and operations are performed using registers.
* The instruction specifies the register(s) involved in the operation.

**Syntax:**  
MOV A, Rn (where n = 0 to 7)

am3

**Examples:**

* MOV A, B → Copy the value from register **B** into register **A**.
* ADD B → Add the value of register **B** to **A** and store the result in **A**.
* INR A → Increment the value of register **A** by **1**.

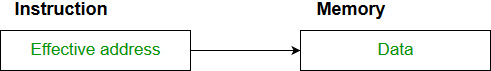
**Use Case:** Performing arithmetic/logical operations between registers.

**3. Direct Addressing Mode**

**Definition:**

* The memory address of the operand is directly specified in the instruction.
* The instruction fetches data from the specified memory location.

**Syntax:**  
MOV A, addr or MOV addr, A



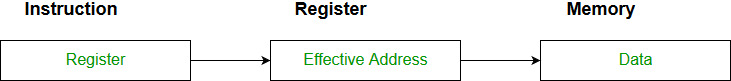
**Examples:**

* MOV A, 30H → Load A from internal RAM at 30H
* MOV 40H, A → Store A into RAM location 40H
* SETB P1.0 → Set Port 1.0 bit directly

**Use Case:** Useful when fetching data directly from a known memory address.

**4. Register Indirect Addressing Mode**

* The operand is stored in **memory**, but the memory address is held in a **register**.
* The effective address is stored in **in R0 or R1 (acting as pointers). Only R0 and R1 are allowed in this mode.**

**Syntax:**  
MOV A, @R0 or MOV @R1, A

**Example:**

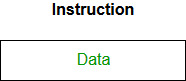
* MOV A, @R0 → Load A from memory location pointed by R0
* MOV @R1, A → Store A into memory pointed by R1

**Use Case:** Useful for working with **arrays, pointers, and tables**.

**5. Implied Addressing Mode**

* The operand is **implicitly specified** in the instruction itself.
* The data is either **8-bit or 16-bit** and is part of the instruction.
* This mode is mainly used for instructions that **do not require an explicit operand**.

**Syntax:**  
CLR A, CPL A, DA A



**Example:** CLR A → Clear the accumulator

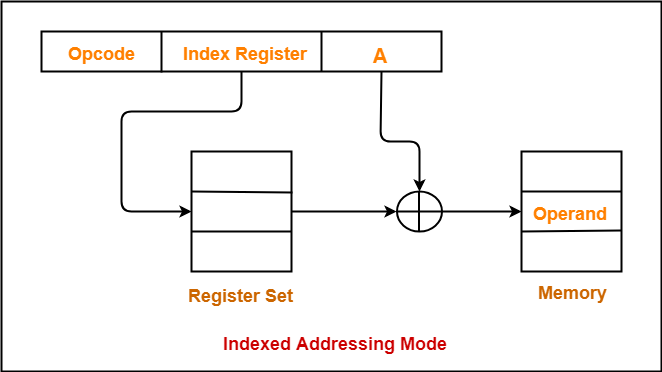
**Use Case:**

**Use Case:** Used in flag operations, accumulator-based instructions, and stack operations.

**6. Indexed Addressing Mode**

* Indexed addressing is a memory addressing mode where the effective address of an operand is calculated by adding a constant value (offset) to the contents of an index register.
* Used to access **code memory (ROM)**. The effective address is formed by **adding A (accumulator)** to **DPTR or PC**.

**Syntax:**  
MOVC A, @A+DPTR or MOVC A, @A+PC



**Examples:**

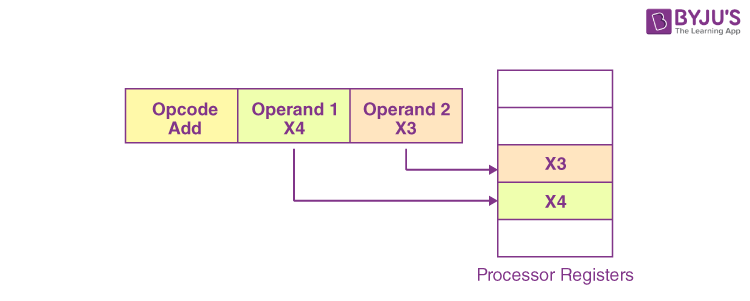
* MOVC A, @A+DPTR → Fetch byte from ROM at DPTR + A
* MOVC A, @A+PC → Fetch byte from ROM at PC + A

**Use Case:**

* Reading lookup tables, constant arrays, fonts, etc.

**7. Register-Specific Addressing Mode**

**Definition:**  
8051 contains 4 banks of registers: Bank 0 to Bank 3, each with R0 to R7. The **Program Status Word (PSW)** determines the **active bank**.



**Examples:**

* MOV A, R3 → Depending on active bank, R3 could be at different RAM address
* ADD A, R5
* INC R7

**Use Case:** Switching banks for interrupt handling or nested functions

**8. Stack Addressing Mode**

**Definition:** The **stack** is a special area of RAM used for temporary storage during subroutine calls and interrupts. Stack is accessed using the **Stack Pointer (SP)** implicitly.

**Examples:**

* PUSH 0E → Push contents of 0EH to stack
* POP 0E → Pop value from stack to 0EH
* CALL 2000H → Push current PC on stack, jump to 2000H
* RET → Pop PC from stack and return to caller

**Use Case:**

* Function call return addresses
* Saving temporary values during execution
* Handling interrupts